WHAT IS CLAIMED IS:

1. A method of implementing a mutual exclusion lock, the mutual exclusion lock capable of preventing at least one acquiring process from operating on at least one shared data object, the at least one acquiring process identified by at least one acquiring process ID, the mutual exclusion lock including at least one variable capable of storing the at least one acquiring process ID, wherein only the at least one acquiring process identified by the at least one acquiring process ID stored in the at least one variable can operate on the at least one shared data object, comprising the steps of:

(a) determining whether the mutual exclusion lock is clear;

(b) if the mutual exclusion lock is clear,

storing the at least one acquiring process ID to the at least one variable, and

operating on the at least one shared data object; and

(c) if the mutual exclusion lock is not clear and if at least one old process identified by at least old process ID stored in the at least one variable included in the mutual exclusion lock is dead,

assigning the at least one acquiring process ID to the at least one variable,

performing a recovery mechanism to recover the at least one data object to a consistent state, and

operating on the at least one shared data object.

2. The method of claim 1, further comprising the steps of:

if the mutual exclusion lock is not clear and if an old process identified by an old process ID stored in the at least one variable included in the mutual exclusion lock is alive, repeat the steps (a)-(c).

3. The method of claim 1, wherein step (b) further comprises the step of clearing the mutual exclusion lock.

4. The method of claim 3, wherein the step of clearing the mutual exclusion lock comprises storing a clear value in the at least one variable in the mutual exclusion lock.

5. The method of claim 1, wherein step (c) further comprises the steps of:

resetting the recovery mechanism, and

clearing the mutual exclusion lock.

6. The method of claim 1, further comprising querying a programming environment to determine whether an old process is dead or alive.

7. The method of claim 6, wherein the frequence -25-of querying a programming environment is limited by an upper bound.

8. The method of claim 5, wherein the step of clearing the mutual exclusion lock comprises storing a clear value in the at least one variable in the mutual exclusion lock.

9. A method of implementing a mutual exclusion lock, the mutual exclusion lock capable of preventing at least one acquiring process from operating on at least one shared data object, the at least one acquiring process identified by at least one acquiring process ID, the mutual exclusion lock including at least one variable capable of storing the at least one acquiring process ID, wherein only the at least one acquiring process identified by the at least one acquiring process ID stored in the at least one variable can operate on the at least one shared data object, comprising the steps of:

comparing the at least one variable and a clear value;

if the at least one variable is equal to the clear value,

storing the acquiring process ID in the at least one variable,

operating on the at least one shared data object, and

writing the clear value to the at least one variable;

if the at least one variable is not equal to the clear value,

querying a programming environment if at least one old process identified by at least one old process ID is dead, wherein the at least one old process ID is equal to the at least one variable,

if the at least one old process is dead,

comparing the at least one variable and the at least one old process ID,

if the at least one variable is equal to the at least one old process ID,

storing the acquiring process ID in the at least one variable,

performing a recovery mechanism to recover the at least

one data object to a consistent state,

operating on the at least one shared data object,

resetting the recovery mechanism, and

writing the clear value to the at least one variable;

wherein the steps of comparing the at least one variable and the clear value and storing the at least one acquiring process ID in the at least one variable if the at least one variable is equal to the clear value are performed atomically; and

wherein the steps of comparing the at least one variable and the at least one old process ID and storing the at least one acquiring process ID in the at least one variable if the at least one old process is dead are performed atomically.

10. The method of claim 9, wherein the clear value is NULL.

11. The method of claim 9, wherein the steps of comparing the at least one variable and the clear value and storing the at least one acquiring process ID in the at least one variable if the at least one variable is equal to the clear value are performed in an atomic Compare-and-Swap operation.

12. The method of claim 9, wherein the steps of comparing the at least one variable and the at least one old process ID and storing the at least one acquiring process ID in the at least one variable if the at least one old process is dead are performed in an atomic Compare-and-Swap operation.

13. The method of claim 9, wherein the step of performing a recovery mechanism to recover the at least one data object to a consistent state comprises reverting the at least one variable to a value at a previous state.

14. The method of claim 9, wherein the frequence of querying a programming environment is limited by an upper bound.

15. A machine-readable medium having instructions stored thereon for execution by a processor to perform a method of implementing a mutual exclusion lock, the mutual exclusion lock capable of preventing at least one acquiring process from operating on at least one shared data object, the at least one acquiring process identified by at least one acquiring process ID, the mutual exclusion lock including at least one variable capable of storing the at least one acquiring process ID, wherein only the at least one acquiring process identified by the at least one acquiring process ID stored in the at least one variable can operate on the at least one shared data object, comprising the steps of:

(a) determining whether the mutual exclusion lock is clear;

(b) if the mutual exclusion lock is clear,

storing the at least one acquiring process ID to the at least one variable, and

operating on the at least one shared data; and

(c) if the mutual exclusion lock is not clear and if at least one old process identified by at least old process ID stored in the at least one variable included in the mutual exclusion lock is dead,

assigning the at least one acquiring process ID to the at least one variable,

performing a recovery mechanism to recover the at least one data object to a consistent state, and

operating on the at least one shared data object.